Phadeo : un environnement pour FPGA virtuel

Sebastián Tleye, Ciprian Teodorov, Erwan Fabiani, Loic Lagadec

▶ **To cite this version:**

Sebastián Tleye, Ciprian Teodorov, Erwan Fabiani, Loic Lagadec. Phadeo : un environnement pour FPGA virtuel. 2015. hal-01179474

HAL Id: hal-01179474
https://hal.univ-brest.fr/hal-01179474

Preprint submitted on 22 Jul 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ComPAS'2015 : Parallélisme/ Architecture / Système
Lille, France, du 1 au 3 juillet 2015

Phadeo : un environnement pour FPGA virtuel

Sebastián Tleye, Ciprian Teodorov, Erwan Fabiani, Loïc Lagadec

Lab-STICC UMR 6532 - MOCS
ENSTA Bretagne,
2 rue Francois Verny,
29806 Brest - France
loic.lagadec@ensta-bretagne.fr

Résumé
Les FPGAs virtuels présentent de nombreux avantages : portabilité du bitstream, gestion de l'obsolescence, etc. acquis au prix d'une perte en performances (surface, fréquence) et de la mise au point d'un environnement d'exploitaiton adéquat. Cet article présente Phadeo, un environnement issu de Madeo, développé dans le cadre du projet ANR ARDyT, qui permet d'évaluer les architectures reconfigurables au travers d'outils d'exploitation. Phadeo est développé en Pharo [1], un langage et environnement inspiré de Smalltalk, et possède de bonnes caractéristiques telles que l'extensibilité ou la modification

Mots-clés : FPGA, virtualisation, prototypage virtuel

---

1. Introduction

La notion de FPGA virtuel [6] trouve son utilité à la fois en amont de la réalisation de dispositifs, lors d'une phase à vocation innovante ou pré-industrielle, et en aval de la rupture de disponibilité. Dans le cadre du projet ARDyT [2], plusieurs architectures reconfigurables alternatives sont étudiées. Ces architectures concourent à une meilleure résilience aux pannes et défauts, sans recours à un durcissement technologique. Ces architectures requièrent en revanche des outils logiciels pour les programmer lesquels fournissent une qualification des architectures.
Les outils permettant d'exploiter une architecture reconfigurable sont structurés en couches. A haut niveau, on peut par exemple retrouver des outils de synthèse de haut niveau (HLS), puis une phase de synthèse architecturale (codage RTL), de synthèse logique (codage en porte), de synthèse physique (allocation de ressources).
Ces différentes étapes doivent pouvoir être menées de façon indépendante, dans des environnements différents de façon à faire émerger rapidement une chaine d'outils. En particulier, des travaux antérieurs doivent pouvoir être intégrés, à minima comme des solutions temporaires devant être remplacées par la suite. Plus on descend dans cette chaine - et donc on se rapproche de la synthèse physique - plus on est dépendant de la cible technologique. Ceci introduit une complexité croissante, qui justifie que la synthèse physique se décompose elle même en plusieurs sous problèmes distincts : l'agencement spacial, le placement, le routage, etc.
Cet article présente Phadeo, un outil de synthèse physique extensible et modifiable pour FPGA, et montre comment les applications et les architectures sont modélisés, quels sont les algorithmes utilisés pour la synthèse physique et comment il est possible d'étendre facilement le

système de toutes nouvelles architectures sans affecter les fonctionnalités existantes.
Nous allons montrer aussi comment des architectures créées dans Madeo [5] peuvent être utilisées dans Phadeo en les exportant avec Fame [4]. Ce faisant, nous allons profiter de la capacité de Madeo à créer des architectures sans avoir à la ré-implémenter dans Phadeo.
Le reste du papier est structuré comme suit : La section 2 décrit les modèles existants (application, architectures) et leurs outils associés.
La section 3 introduit une différene majeure des outils par rapport à Madeo, qui consiste en un passage à l'échelle des algorithmes en offrant des projections contextuelles des attibuts utiles de structures de données complexes.
La section 4 offre une vision synthétique du système logiciel au travers de certains indicateurs connus, comme la pyramide suggérée par Michele Lanza et Radu Marinescu dans leur livre Metrics orientées objet dans la pratique [8] . Nous allons montrer quelques repères créés pour mesurer le placement et le temps de routage.
Enfin, dans la section **??**, nous allons parler des avantages et inconvénients de Phadeo et les travaux futurs nécessaires.

2. Outils existants et interopérabilité

2.1. Modèles

Les outils de synthèse physique permettent de réaliser l'allocation de ressources (essentiellement le placement et le routage) en aval de la synthèse d'architecture. Ces couches basses sont intégrées dans les outils logiciels offerts par les fournisseurs de solutions matérielles (par exemple ISE dans le cas de Xilinx). En revanche, lors de la définition de nouvelles architectures, il convient de s'appuyer sur des outils plus génériques, c'est-à-dire sachant s'adapter à des architectures ouvertes, paramétriques, etc. Parmi ces outils, on retrouve VPR [3] et Madeo [7]. VPR est un outil largement répandu dans la communauté académique, qui permet de manipuler des architectures régulières avec de bonnes performances. Madeo, est un environnement moins performant mais plus flexible et ouvert. Il permet en particulier de considérer une plus grande variété d'architectures. Actuellement Phadeo supporte deux méta-modèles d'architectures, correspondant aux architectures MADEO et VPR.
Pour garantir la flexibilité dans Phadeo, de façon à pouvoir y ajouter n'importe quel type d'architecture, **FPGAArchitecture** possède un comportement minimal comme illustré par la [Figure 1].
Les architectures Madeo sont les architectures utilisées par l'outil Madeo, elles sont modélisées dans Phadeo avec une composition de **RaObjet**s et instances de la classe **FPGAMadeoArchitecture**. Les **RaObjects** sont les instances de la hiérarchie de classe d'architecture définie dans Madeo.

2.2. Outils

Les architectures sont une composition complexe d'objets Smalltalk,conforme à un méta-modèle. La mise en œuvre du modèle pour créer ces architectures demande des constructeurs, sachant agglomérer ces instances de façon cohérente. Ceci est complexe à assurer. Deux approches opposées sont retenues en fonction du métamodèle.
Pour les architectures Madeo, la solution retenue consiste à créer les architectures dans Madeo puis les exporter vers un fichier **.mse** utilisant Fame [4]. Le choix a donc été fait de privilégier l'interopérabilité au dépend du développement d'environnements complets.
Pour les architectures VPR, le choix est fait d'offrir des constructeurs dans l'environnement

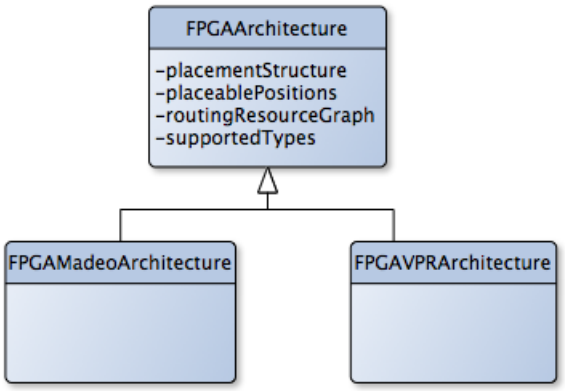

FIG. 1 – La classe FPGAArchitecture possède deux sous classes concrètes : FPGAMadeoArchitecture and FPGAVPRArchitecture

Phadeo. En contrepartie, seul un sous ensemble du métamodèle VPR est considéré. Ces architectures sont définies dans un fichier XML qui contient toutes les informations nécessaires, tels que des ports, les switchs, les canaux, etc. Phadeo lit les fichiers XML et instancie toutes les classes du modèle d'architecture de VPR.
Phadeo possède sa propre hiérarchie de classes, dans laquelle le constructeur de l'architecture est la classe **FPGAVPRArchitecture**qui crée des instances de architectures VPR en recevant un fichier XML comme argument, le format de ce XML est défini dans le manuel de VPR. Une architecture VPR se compose d'instances de **FPGAChannel**, **FPGAVPRPin**, **FPGASwitchBlock**, etc. La [Figure 2] illustre l'articulation entre les environnements Madeo et Phadeo. Dans Phadeo, on retrouve le même méta-modèle dont les classes sont intanciées par Fame lors de la lecture du fichier **.mse**.

2.3. Interopérabilité

Fame est un framework de méta-modélisation à l'exécution. Il est utilisé pour la sérialisation (lecture / écriture) d'objets, l'échange d'objets et de modèles entre applications. Il en existe des implémentations pour de nombreux langages comme pour Squeak Smalltalk, Java, Python, Cincom Smalltalk, Pharo Smalltalk, etc.
Dans ce contexte, l'objectif est d'exporter la composition des objets qui modélisent l'architecture de Madeo et de les importer dans Phadeo. Pour ce faire, toutes les classes dont les instances sont exportables sont annotées. Dans Madeo (Cincom Smalltalk), la classe RaObject et toutes ses sous-classes sont annotées, pour ce faire les classes doivent mettre en œuvre la méthode **fameAnnotation**avec le nom de la classe, la superclasse et le paquet. Le code ci-dessous,illustre l'annotation Fame de la classe **RaObject**.

```
fameAnnotation
  <MSEClass: #RaObject super: #Object>
  <package: 'Ra'>
```

Une fois que toutes les classes sont annotées et les objets sont créés, un code effectue une itération sur les objets et crée un fichier sérialisé (en utilisant Fame). Ce fichier, qui porte une extension **.mse**, est relu par Fame du le côté Phadeo.
Dans Phadeo, une application (ou une description de circuit) est modélisée par un graphe dont

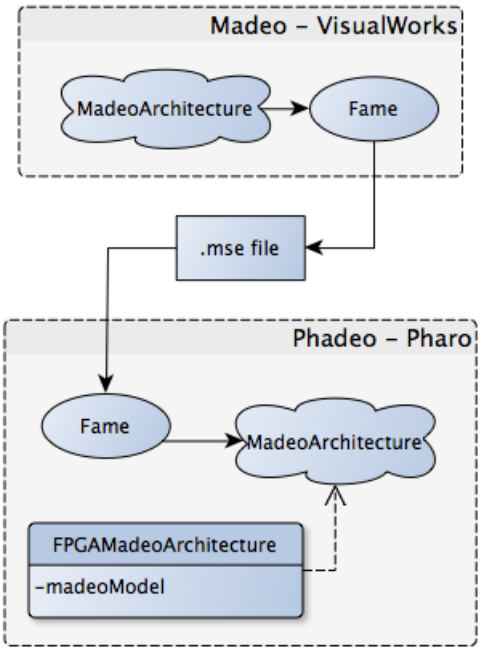

FIG. 2 – Les différents pas dans l'exportation des architectures Madeo

les nœuds sont les blocs logiques et les arcs sont les liens entre ces blocs logiques. Les applications sont normalement créées après lecture d'un fichier au format BLIF. Phadeo analyse le fichier à la recherche des entrées, les sorties et les neouds intermédiaires. Chaque noeud dans le graphe est typé, les entrées et les sorties auront pour Type **IO** tandis que les nœuds restants auront pour Type **CLB**. Typer les nœuds permet de contraindre l'algorithme de placement, puisqu'il n'est pas possible de placer n'importe quel bloc logique dans n'importe quelle cellule de l'architecture.

3. Refonte logicielle

L'un des objectifs de Phadeo est de gérer différents types d'architectures, Phadeo met donc en œuvre des structures de données qui font abstraction de l'architecture, et offrent un support spécifique à chaque opération : placement, routage, etc. Ces structures sont instanciées par un visiteur qui traverse tous les éléments de l'architecture . Ce visiteur n'est dépendant que du méta modèle, et il n'est donc pas nécessaire de le redévelopper pour un nouveau modèle. En revanche, une évolution du méta-modèle requiert de reprendre le code du visiteur (typiquement ajouter une nouvelle méthode de visite pour un nouvel élément de métamodèle).
Les structures sont manipulées par des algorithmes de synthèse physique [9]. La structure de placement maintient les informations de positions libres, de cellules occupés et les types autorisés par une position.
La classe de la structure de placement possède un dictionnaire des dictionnaires [Figure **??**], dont les clés sont les types extraits de l'architecture, et les valeurs sont des dictionnaires avec tous les éléments et les positions du type correspondant.
Les positions dans la structure de positionnement supportent un référencement symbolique, ce qui permet de s'adapter à toute topologie architecturale. Une fois que tous les emplacements

pour les blocs logiques dans un circuit ont été choisis, le routeur détermine quelles ressources de routage sont mobilisées pour implémenter l'interconnexion. Pour ce faire, un graphe orienté est extrait de l'architecture dans laquelle chaque ressource devient un noeud et les connexions possibles entre ressources deviennent des arcs.

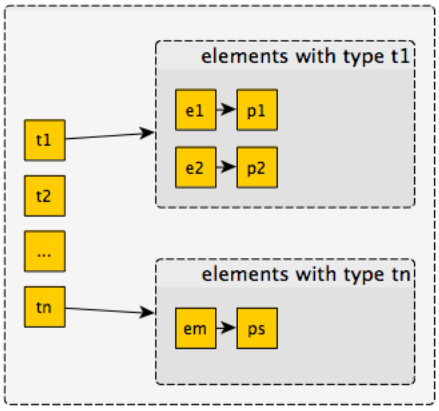

FIG. 3 – Le placement repose sur un dictionnaire de dictionnaires, mémorisant les positions possibles en fonction du type des ressources

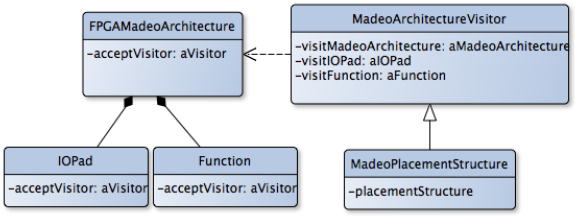

FIG. 4 – MadeoPlacementStructure est un visiteur concret sur les architectures Madeo

Le graphe ressources de routage extrait, est un graphe orienté où tous les nœuds correspondent aux entrées et sorties dans l'architecture, et les arcs correspondent à des connexions possibles entre ces nœuds.
Dans l'exemple fourni par Figure 6 tous les canaux et toutes les E/S sont devenues un nœud dans le graphe de ressources de routage. Input1 et input2 sont connectés à wire1 et wire2 respectivement, ce qui signifie que dans le graphique des ressources de routage, il y aura une arête entre les nœuds qui représentent input1, input2 et les nœuds qui représentent wire1 et wire2. L'aspect directionnel ou non des connexions se traduit par un possible doublement des arcs dans le graphe de routage (par exemple entre wire1 et wire 3). Le graphe est ensuite parcouru par un algorithme A*.

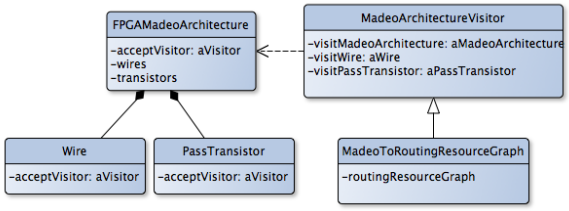

FIG. 5 – MadeoToRoutingResourceGraph est un visiteur sur les architectures Madeo. Il produit le graphe des ressources de routage

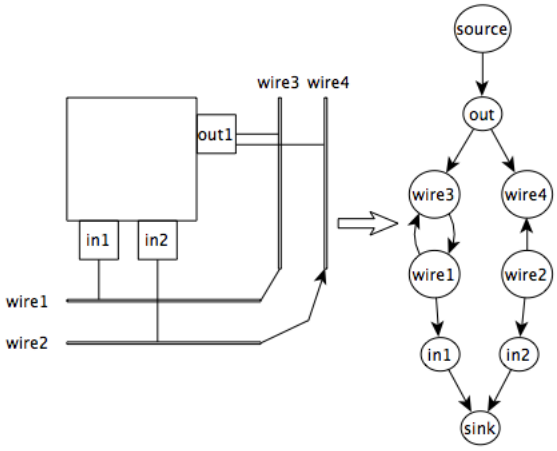

FIG. 6 – Le graphe acyclique produit par l'architecture

4. Optimisations

Comme nous l'avons mentionné, les points forts de Phadeo sont l'extensibilité et modifiabilité du framework. Mesurer ces critères est difficile, mais peut cependant reposer sur des concepts tels que le masquage d'information, l'encapsulation, le polymorphisme, la cohésion, le respect de principes de conception, etc.
La [Figure 7] représente la pyramide utilisée by Michel Lanza and Radu Marinescu dans leur livre Object-Oriented Metrics in Practice [8]. Elle montre la taille et la complexité du système, et diverses métriques :

1. size and complexity ratios
   (a) CYCLO (Intrinsic Operation Complexity) : Le nombre de branchements conditionnel par ligne de code
   (b) LOC (Operation Structure) : Le nombre de ligne de code par méthode
   (c) NOM (Class structure) : Le nombre de méthodes par classe
   (d) NOC (High-Level Structuring) : Le nombre de classes par paquets
2. Coupling
   (a) CALL (Coupling intensity) : Le nombre d'invocations divisé par le nombre de méthodes

(b) FANOUT (Coupling dispersion) : Le nombre de classes divisées par le nombre d'invocations

3. Hierarchy
   (a) ANDC : Le nombre moyen de classes dérivées
   (b) AHH : La profondeur moyenne de la hiérarchie de classes

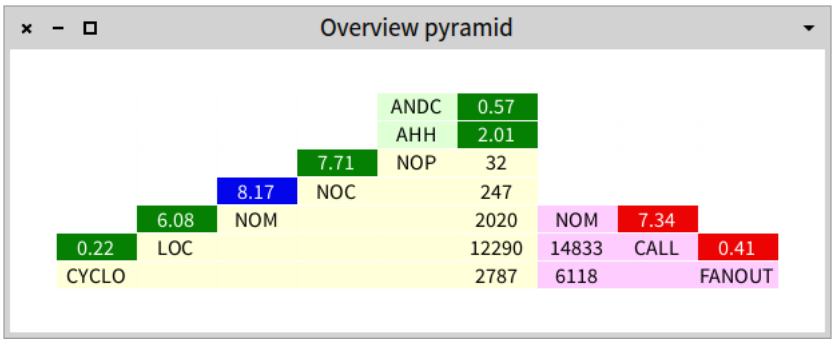

FIG. 7 – La pyramide d'évaluation suggérée par Lanza dans [8]

La [Figure 7] montre les valeurs vertes, bleues et rouges, vert signifie que la valeur est une valeur moyenne de systèmes Smalltalk, rouge qui est supérieur à la moyenne et le bleu est inférieur à la moyenne. Basse, moyenne et des valeurs élevées ont été calculés statistiquement de plus de 20 systèmes différents Smalltalk, ceci est expliqué dans [10].

La représentation en araigné repose sur un code de couleurs où les bornes min et max sont respectivemetn en bleu et rouge.
La figure 8 montre que Madeo est caractérisé par des valeurs élevées de Operation Structure, Class Structure, High level structuring, et coupling intensity. Ces valeurs traduisent une conception en phase avec les pratiques des années 2000.
Pour Phadeo, les valeurs sont proches des standards Smalltalk, comme illustré par la figure 9. Cela suggère que Phadeo est d'avantage conforme aux pratiques de conception/codage smalltalk, et donc exhibe des carctéristiques fondamentales de ces systèmes, dont l'extensibilité.
Phadeo exhibe cependant des valeurs élevées pour CALL et FANOUT, ce qui suggère que l'opération d'appel ne met pas en relation les bons interlocuteurs. Ceci s'explique facilement, car Phadeo fait un usage massif de délégation dans ses méthodes, la délégation provoque l'augmentation des invocations par méthode, par conséquent, des valeurs élevées de CALL.Cependant, la délégation permet à Phadeo d'être extensible pour considérer des architectures qui peuvent être complètement différentes de celles qui existent actuellement.Comme aucune hypothèse ne doit être fairte sur la nature de ces futures architectures, il convient de déléguer les responsabilités.
La conception de Phadeo permet une bonne agilité lors du développement. En particulier, l'approche modulaire permet de cerner les points chauds, candidats pour une optimisation. L'optimisation peut prendre deux formes principales. La première consiste à améliorer le design de la solution, avec pour effet une possible diffusion des gains lors de plusieurs phases de la synthèse. C'est par exemple le cas lors de l'optimisation des classes techniques, comme celle

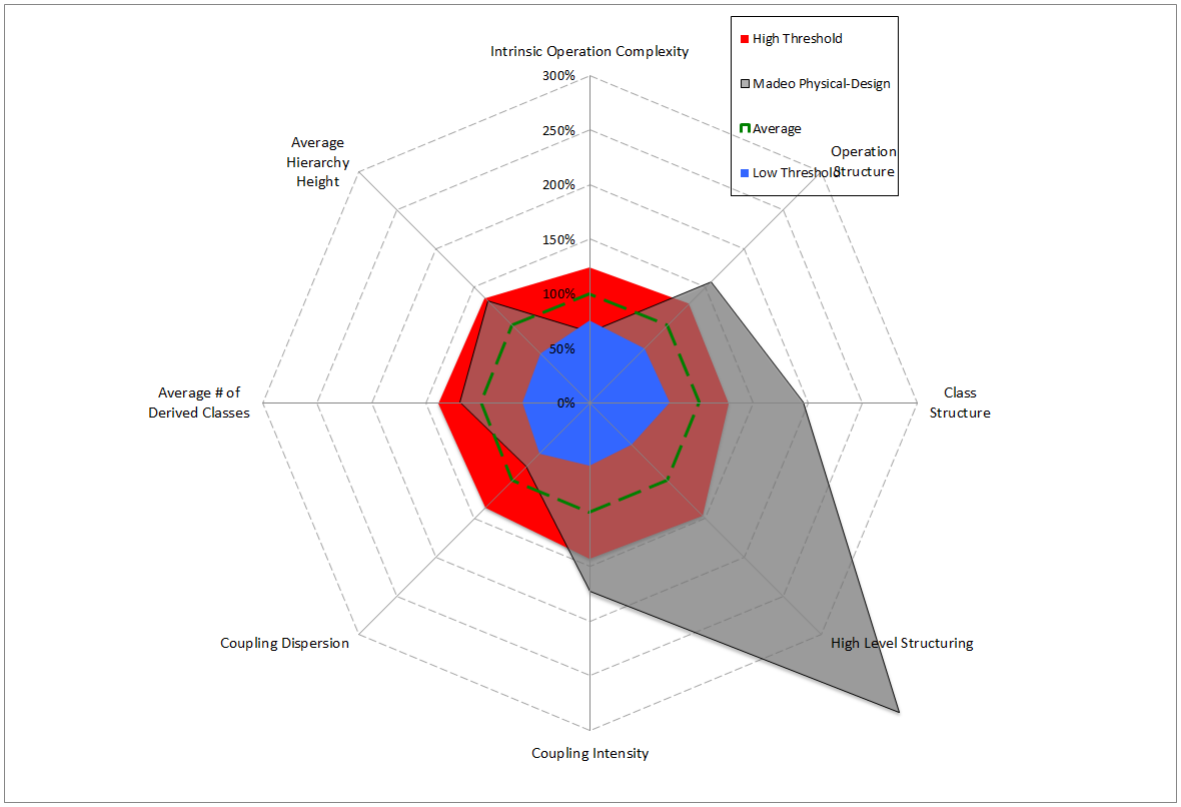

FIG. 8 – Madeo comparé à 20 systèmes Smalltalk

implémentant le recuit simulé, que l'on mobilise lors du floorplanner, du packing et du placement. La seconde consiste à privilégier l'efficacité de la solution quite à dégrader la lisibilité du design.

Comme le montre la [Figure 10], une diminution drastique de temps de placement a été obtenu pendant le développement. Sur cette figure, l'axe des abscisses représente le temps, et en particulier les dates des auxquelles les musures ont été réalisées. En Y, on retrouve le temps de traitement. La réuction du temps de placement a été causée par une amélioration de l'algorithme du recuit simulé. L'algorithme du recuit simulé génère de nouveaux états en regardant pour les voisins de l'état actuel, puis, selon un calcul de probabilité, décide si elle accepte le nouvel état ou non. si l'état est accepté, il remplace (est copié dans) l'état actuel. Le problème avec cette approche était que la copie des états affectait toute la structure d'optimisation (qui comprend la structure de placement). Cette copie s'avérait couteuse car elle portait sur des graphes et générait donc plusieurs parcours sur les structures.

La solution retenue est de mettre en place des opérations inverses. Il n'est plus nécessaire de copier les états, et la structure manipulée est unique. Chaque changement est implicitement validé, en cas de rejet du nouvel état, l'opération inverse est portée sur la structure. Seul l'optimal est copié (car le recuit permet de retenir des solutions sous optimal pour sortir de minima locaux).

L'évolution des performances du routage, représenté par la [Figure 11], montre un tracé relativement similaire. Pour accélérer l'algorithme, Le visiteur a été ré-écrit. Le maintien des noeuds visités n'est plus du ressort du visiteur (qui possédait une collection de noeuds) mais de chaque

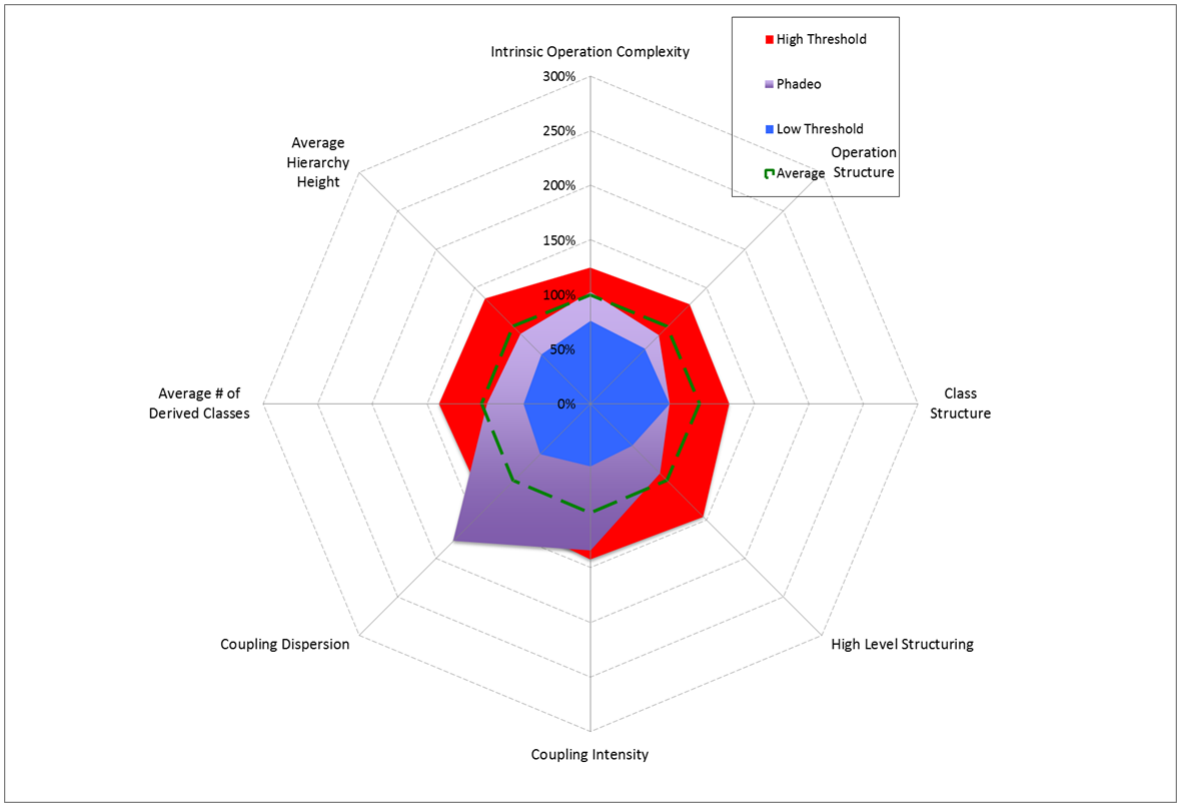

FIG. 9 – Phadeo comparés à 20 systèmes Smalltalk

noeud (qui possède maintenant une variable d'instance booléenne). Cela allège le besoin de parcours de collection, couteux en temps. Avoir cette variable d'instance dans les noeud n'est pas une solution élégante en terme de conception. En effet cela consiste à modifier le modèle en fonction des besoins des outils, ce qui s'apparente à une pollution dans une approche d'ingénierie dirigée par les modèles. En revanche, cela augmente la vitesse de près de 50 %.

5. Conclusion et perspectives

Dans cet article, nous avons introduit Phadeo, un nouvel outil de synthèse physique basée sur Madeo. Nous montrons comment les applications sont modélisées et comment elles sont créées à partir de fichiers BLIF. Cet article introduit également les deux métamodèles d'architecture supportés actuellement : Madeo et VPR.

Nous montrons un aperçu du modèle de classes et comment il est possible d'étendre Phadeo avec de nouvelles architectures, sans la nécessité de modifier l'une des classes existantes (principe d'ouverture fermeture). C'est le point fort de Phadeo et très important dans le contexte dans lequel nous travaillons.

Nous avons expliqué la mise en place des traitemetns de la synthèse physique. Cela repose sur les des serveices transverse comme le recuit simulé, et l'emploi de structures de données extraites automatiquement du modèle en fonction des traitements à réaliser par les outils.

Dans un second temps, Phadeo a été caractérisé au travers de diverses métriques. Phadeo a ainsi plus de couplage qu'un système Smalltalk ordinaire, mais nous soutenons que ce n'est pas un problème puisque ce couplage vient de l'utilisation de la délégation. Nous avons également

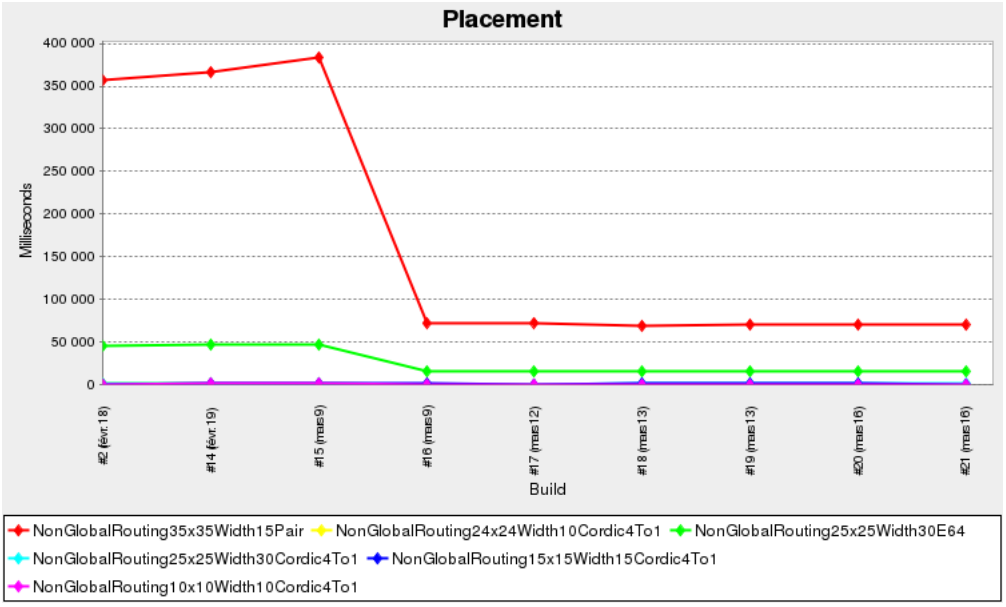

FIG. 10 – Benchmarks de placement pour des architectures Madeo

vu comment une simple modification dans l'algorithme du recuit simulé améliore la vitesse près de quatre fois. Dans l'algorithme de routage, nous avons décidé de réduire une bonne conception pour améliorer la vitesse, ce qui est normal dans les grands systèmes, le compromis entre la performance et le design est toujours présent.
Comme il est toujours en cours de développement, Phadeo a quelques limitations et certaines fonctionnalités manquent. En particulier, le floorplanning, et la génération de bitstreamer. Phadeo n'a pas d'interface utilisateur conviviale. Cette limitation en interdit l'usage par des tiers dans l'état actuel. En revanche, la mise en place de chaine d'outils, et donc la capacité à exploiter les outils de visualisation de Madeo, est fonctionnelle.
Les prochaines étapes concernent maintenant l'adaptation des outils à des architectures spécifiques, en cours de définition avec les partenaires de ARDyT. Ces architectures exhiberont des caractéristiques atypiques, que saura exploiter Phadeo, car il est à la fois suffisamment flexible pour intégrer de nouveaux élements architecturaux et outils, et suffisamment performant pour envisager un passage à l'échelle.

Remerciements

Le projet ARDyT est financé par l'Agence Nationale pour la Recherche française (ANR) sous la référence ANR-11-INSE-015.

Bibliographie

1. Pharo. – http ://www.pharo-project.org/.
2. ArdyT consortium. – http ://www.ardyt.irisa.fr.
3. Betz (V.) et Rose (J.). – Vpr : A new packing, placement and routing tool for fpga research. – In *Field Programmable Logic and Application*, *LNCS*, LNCS, 1997.
4. Kuhn (A.) et Verwaest (T.). – Fame, a polyglot library for metamodeling at runtime. – In *In Workshop on Models at Runtime*, pp. 57–66, 2008.

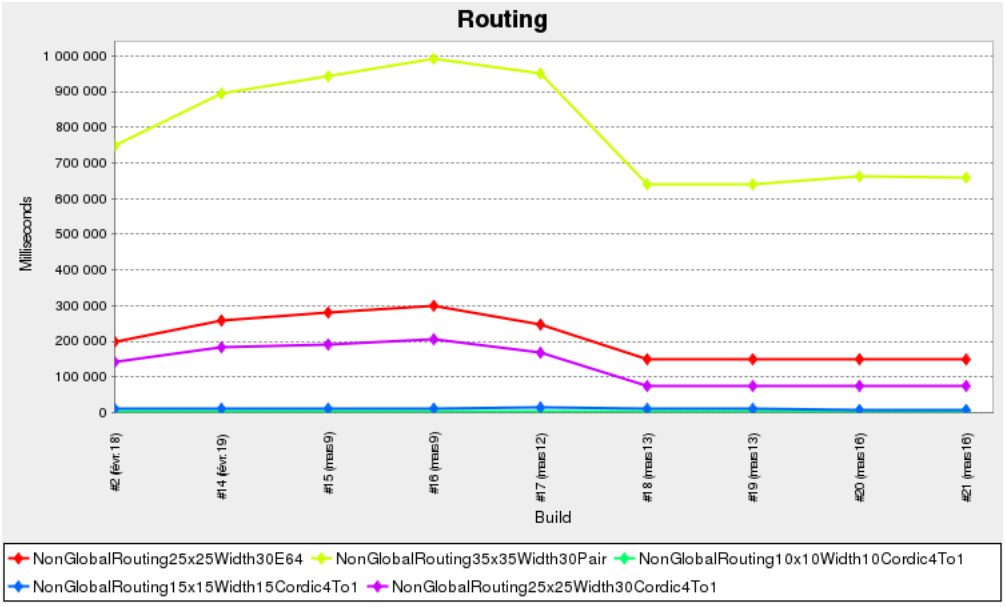

FIG. 11 – Benchmark de routage sur des architectures Madeo

5. Lagadec (L.). – *Abstraction, modelisation et outils de cao pour les architectures reconfigurables*. – Thèse de PhD, Université de Rennes I, 2000.
6. Lagadec (L.), Le Lann (J.-C.) et Bollengier (T.). – A Prototyping Platform for Virtual Reconfigurable Units. – In *RECOSOC 2014*, p. xx, Montpellier, France, mai 2014.
7. Lagadec (L.), Teodorov (C.), Lann (J.-C. L.), Picard (D.) et Fabiani (E.). – Model-driven toolset for embedded reconfigurable cores : Flexible prototyping and software-like debugging. 2014.
8. Lanza (M.), Ducasse (S.) et Marinescu (R.). – *Object-Oriented Metrics in Practice : Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. – Springer Berlin Heidelberg, 2007.
9. McMurchie (L.) et Ebeling (C.). – Pathfinder : A negotiation-based performance-driven router for fpgas. – In *Proceedings of the 1995 ACM Third International Symposium on Field-programmable Gate Arrays*, *FPGA '95*, FPGA '95, pp. 111–117, New York, NY, USA, 1995. ACM.
10. Teodorov (C.) et Lagadec (L.). – Model-driven physical-design automation for fpgas : fast prototyping and legacy reuse. *Software : Practice and Experience*, vol. 44, n4, mars 2013, pp. 455–482. – WOS.